

PLP - Segundo Parcial - 1^{er} cuatrimestre de 2023

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R). Las notas para cada ejercicio son: -, I, R, B-, B. Entregar cada ejercicio en hojas separadas. Poner nombre, apellido y número de orden en todas las hojas, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras. El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolverlo.

Ejercicio 1 - Programación Lógica

- a) Sea un alfabeto un conjunto de átomos cualesquiera, y sea una palabra en cierto alfabeto A una lista finita de átomos que pertenecen a A. Implementar el predicado palabra(+A, +N, -P), que genera las palabras en A cuya longitud es de exactamente N elementos. Tanto en este inciso como en el siguiente, no es obligatorio respetar el mismo orden de los ejemplos.

```
?- palabra([a,b,c], 4, P).  
P = [a,a,a,a] ;  
P = [a,a,a,b] ;  
...  
P = [b,c,a,b] ;  
...  
P = [c,c,c,c] ;  
false
```

- b) Sea una frase en cierto alfabeto A una lista finita de palabras no vacías en A. Implementar el predicado frase(+A, -F), que genera todas las posibles frases en A.

```
?- frase([q,u,e,s,o], F).  
F = [] ;  
...  
F = [[q,u,e],[e,s],[e,s,o]] ;  
...  
F = [[q,u,e,s,o,s]] ;  
...
```

- c) Indicar si el parámetro P del predicado palabra y el parámetro F del predicado frase son reversibles. Justificar la respuesta.

Ejercicio 2 - Resolución

- a) Representar en forma clausal las siguientes fórmulas de lógica de primer orden referidas a números naturales:

- I. $\forall X \forall Y (X = Y) \supset divide(X, Y) \wedge divide(Y, X)$ - Si dos números son iguales, se dividen mutuamente.
- II. $\forall X \forall Y \forall Z (divide(X, Y) \wedge divide(Y, Z)) \supset divide(X, Z)$ - La relación de divisibilidad es transitiva.

- b) Usando el método de resolución demostrar la siguiente fórmula:

$$\forall X \forall Y (X = Y) \supset divide(X, X)$$

- c) La resolución realizada en el punto anterior, ¿fue SLD? Justificar. Si no lo fue, ¿sería posible encontrar una resolución SLD para este conjunto de cláusulas? (No es necesario escribirla, solamente indicar por qué se puede saber que es posible o que no lo es.)

Ejercicio 3 - Objetos

- a) Definir en Cálculo ς la clase *Pokemon*, cuyas instancias puedan responder a los mensajes *salud*, *atacar* y *evolucionar*. La salud inicial de los pokemons es 100, al atacar reducen en 1 la salud de su oponente y al evolucionar, su ataque se incrementa en 1, es decir, cuando un pokemon evoluciona, hace 1 más de daño a su oponente que antes de evolucionar. Tener en cuenta que los pokemons pueden evolucionar más de una vez.
- b) Se definen los siguientes objetos:

$$\begin{aligned} \text{ash} &\stackrel{\text{def}}{=} [\text{cantidadDePokemons} \doteq \text{cero}, \\ &\quad \text{adoptar} = \varsigma(z)\lambda(x)z.\text{cantidadDePokemons} := z.\text{cantidadDePokemons.succ}] \\ \text{pikachu} &\stackrel{\text{def}}{=} \text{Pokemon.new} \end{aligned}$$

cero es el objeto definido en clase, y se tiene que $\text{cero.succ} \longrightarrow \text{uno}'$. Mostrar cómo reduce $\text{ash.adoptar}(\text{pikachu})$. Está permitido usar la regla APP vista en clase.

①

Corrigió: Ignacio

a) % PALABRA (+A, +N, -P)

PALABRA(A, 1, [X]) :- MEMBER(X, A). *Sería mejor que el caso base fuera palabra(A, 0, [X])*

PALABRA(A, N, [X|XS]) :- LENGTH([X|XS], N), N > 1,

Este llamado a length no es necesario

MEMBER(X, A), N2 IS N-1,

PALABRA(A, N2, XS).

COLOQUIALMENTE, LO QUE PALABRA DICE ES QUE P TIENE LARGO N Y QUE CADA ELEMENTO DE P ES UN SÍMBOLO DEL ALFABETO.

b) % FRASE (+A, -F)

FRASE(A, []). *Aquí ponerse en vez de A*

ASUMO QUE ESTA FRASE ES TRUNCADA ES DECIR: $0 \leq X \leq 0$

FRASE(A, [F|FS]) :- BETWEEN(1, INF, L). *Se puede usar desde*

PALABRADEARGO HASTA(A, L, F)

BETWEEN(0, L, L2)

LENGTH([F|FS], L2), L3 IS L2-1

FRASE2(A, FS, L3, L)

PALABRADEARGO HASTA(A, F, L) :- BETWEEN(1, L, LEN),

PALABRA(A, LEN, F).

LA IDEA ES TENER UN GENERADOR INFINITO, QUE NOS INDIQUE HASTA QUE TAN LARGA PUEDE SER UNA PALABRA DE LA FRASE. LUEGO, LA FRASE PODRÁ TENER UNA LONGITUD ENTRE 0 Y EL LARGO HASTA EL CUAL PUEDE LLEGAR UNA PALABRA DE LA FRASE.

FRASE2(A, [], 0, -).

FRASE2(A, [X|XS], N, L) :- PALABRADEARGO HASTA(X, L, X),

N2 IS N-1, FRASE2(A, XS, N2, L).

c) EL PARÁMETRO P EN PALABRA ES REVERSIBLE,
ESTO SE PUEDE VER YA QUE TANTO LENGTH
COMO MEMBER PUEDEN TENER INSTACIADOS SUS
PARÁMETROS. SI ENTRAMOS EN EL 1ER CASO
MEMBER NO TIENE PROBLEMAS SI [x] ESTÁ INSTACIADO,
Y EN EL SEGUNDO CASO SUCEDE LO MISMO TANTO
CON MEMBER COMO CON LENGTH. ADÉMÁS, NO
TENDRIAMOS PROBLEMAS CON LA LISTA VACÍA, YA QUE
NO UNIFICA C/ EL PRIMER CASO Y EL
SEGUNDO NECESITA $N > 1$. ✓

EL PARÁMETRO F DE FRASE. TAMBIÉN ES INVERSIBLE,
YA QUE PODRIAMOS CHEQUEAR QUE CADA PALABRA SEA
UNA PALABRA DE LENGTHASTA Y VER QUE LO QUE SIGUE ES
UNA FRASE². LO DE PALABRA DE LENGTHASTA FUNCIONA
BIEN XQ PALABRA ES REVERSIBLE EN P. X

No es reversible, el primer between se cuelga instanciando los infinitos valores de L)

②

a)

$$I. \forall x \forall y (x=y) \supset \text{DIVIDE}(x,y) \wedge \text{DIVIDE}(y,x)$$

(

$$\forall x \forall y \neg(x=y) \vee (\text{DIVIDE}(x,y) \wedge \text{DIVIDE}(y,x))$$

(

$$\forall x \forall y (\neg(x=y) \vee \text{DIVIDE}(x,y)) \wedge (\neg(x=y) \vee \text{DIVIDE}(y,x))$$

$$C_1 = \{ \neg(x_1=y_1), \text{DIVIDE}(x_1,y_1) \} \checkmark$$

$$C_2 = \{ \neg(x_2=y_2), \text{DIVIDE}(y_2,x_2) \} \checkmark$$

$$II. \forall x \forall y \forall z (\text{DIVIDE}(x,y) \wedge \text{DIVIDE}(y,z)) \supset \text{DIVIDE}(x,z)$$

$$(\forall x \forall y \forall z \neg \text{DIVIDE}(x,y) \vee \neg \text{DIVIDE}(y,z) \vee \text{DIVIDE}(x,z))$$

$$C_3 = \{ \neg \text{DIVIDE}(x_3,y_3), \neg \text{DIVIDE}(y_3,z_3), \text{DIVIDE}(x_3,z_3) \} \checkmark$$

b) PARA DEMOSTRAR PRIMERO LA NIEGO.

$$\forall x \forall y (x=y) \supset \text{DIVIDE}(x,x)$$

(

$$\forall x \forall y \neg(x=y) \vee \text{DIVIDE}(x,x)$$

$$\neg \forall x \forall y \neg(x=y) \vee \text{DIVIDE}(x,x)$$

(

$$\exists x \neg \forall y \neg(x=y) \vee \text{DIVIDE}(x,x)$$

(

$$\exists x \exists y \neg (\neg(x=y) \vee \text{DIVIDE}(x,x))$$

(

$$\exists x \exists y (x=y) \wedge \neg \text{DIVIDE}(x,x)$$

(

$$\exists y (k_1=y) \wedge \neg \text{DIVIDE}(k_1,k_1)$$

(

$$(k_1=k_2) \wedge \neg \text{DIVIDE}(k_1,k_1)$$

$$C_4 = \{ (k_1=k_2) \} \checkmark$$

$$C_5 = \{ \neg \text{DIVIDE}(k_1,k_2) \} \checkmark$$

PASANDO EN LIMPIO:

$$C_1 = \{ \neg (X_1 = Y_1), \text{DIVIDE}(X_1, Y_1) \}$$

$$C_2 = \{ \neg (X_2 = Y_2), \text{DIVIDE}(Y_2, X_2) \}$$

$$C_3 = \{ \neg \text{DIVIDE}(X_3, Y_3), \neg \text{DIVIDE}(Y_3, Z_3), \text{DIVIDE}(X_3, Z_3) \}$$

$$C_4 = \{ (K_1 = K_2) \}$$

$$C_5 = \{ \neg \text{DIVIDE}(K_1, K_1) \}$$

EL PLAN ES EL SIGUIENTE: DEBE QUE SI DOS NÚMEROS X , Y SON IGUALES SE DIVIDEN MUTUAMENTE Y QUE COMO LA DIVISIBILIDAD ES TRANSITIVA SI X DIVIDE A Y , Y DIVIDE A X . ENTONCES X DIVIDE A X .

$$\begin{array}{l} C_1 \quad C_4 \\ \diagdown \quad | \\ X_1 \leftarrow K_1 \\ Y_1 \leftarrow K_2 \\ C_6 = \{ \text{DIVIDE}(K_1, K_2) \} \end{array}$$

$$\begin{array}{l} C_2 \quad C_4 \\ \diagdown \quad | \\ X_2 \leftarrow K_1 \\ Y_2 \leftarrow K_2 \\ C_7 = \{ \text{DIVIDE}(K_2, K_1) \} \end{array}$$

$$\begin{array}{l} C_3 \quad C_6 \\ \diagdown \quad | \\ X_3 \leftarrow K_1 \\ Y_3 \leftarrow K_2 \end{array}$$

$$C_8 = \{ \neg \text{DIVIDE}(K_2, Z_3), \text{DIVIDE}(K_1, Z_3) \}$$

$$\begin{array}{l} C_7 \quad C_8 \\ \diagdown \quad | \\ Z_3 \leftarrow K_1 \end{array}$$

$$\begin{array}{l} C_5 \quad C_9 = \{ \text{DIVIDE}(K_1, K_1) \} \\ \diagdown \quad | \end{array}$$



c) NO, LA RESOLUCIÓN NO ES SLD. ESTO SE DEBE, POR EJEMPLO, A QUE NO FUE LINEAL. ✓

~~NO ES POSIBLE HACER UNA RESOLUCIÓN SLD A LA QUE NO FUE LINEAL. PORQUE LA C4 SE RECONSTRUYÓ DE C2 Y C3, POR LO QUE NO SE PUEDE RESOLVER SIN RESOLVER C2 Y C3, POR LO QUE NO ES LINEAL.~~

SI, PODRÍA HACERSE UNA RESOLUCIÓN SLD, TENIENDO A C_5 COMO CAUSAL OBS, RESOLVIÉNDOLA CON C_3 , EL RESULTADO CON C_1 , LUEGO CON C_2 Y POR ÚLTIMO 2 VECES CON C_4 . ASÍ LA RESOLUCIÓN ES LINEAL, TIENE SOLA CLAUSULAS DE HORN Y C_5 ES OBJETIVO. YO ENCONTRE OTRO CAMINO QUE ME PARECE MÁS CLARO Y CERCANO A UNA DEMOSTRACIÓN "COMÚN". ✓

③

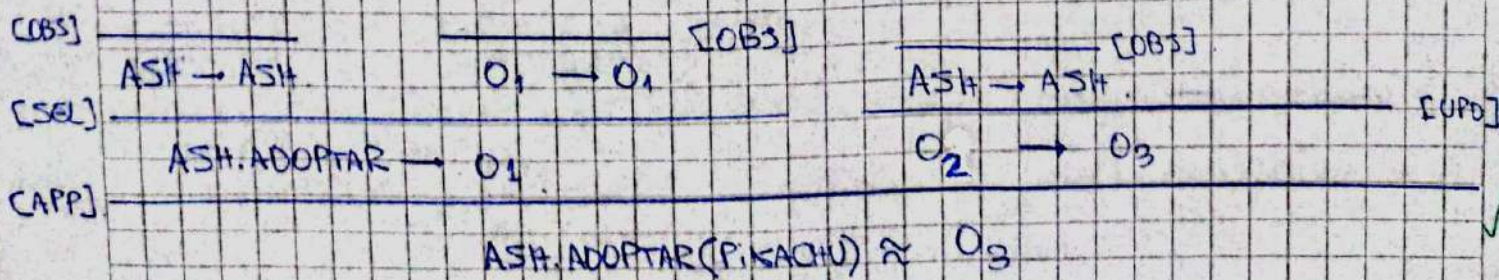
a)

$$\begin{aligned} \text{POKEMON} &\stackrel{\text{def}}{=} [\text{NEW} = C(z) [\text{SAWD} = C(s) z. \text{SAWD}(s), \\ &\quad \text{ATACAR} = C(s) z. \text{ATACAR}(s), \\ &\quad \text{EVOLUCIONAR} = C(s) z. \text{EVOLUCIONAR}(s)], \\ \text{SAWD} &= \lambda(s) 100 \\ \text{ATACAR} &= \lambda(s) \lambda(op) op. \text{SAWD} := op. \text{SAWD} - 1, \\ \text{EVOLUCIONAR} &= \lambda(s) s. \text{ATACAR} := \\ &\quad C(s') \lambda(op') op'. \text{SAWD} := s. \text{ATACAR}(op'). \text{SAWD} - 1. \end{aligned}$$

(no está definido la resta)

LA IDEA DE EVOLUCIONAR ES QUE SE CAMBIA LA MANERA DE ATACAR, DONDE HACEMOS UN ATAQUE AL Oponente como lo haríamos antes de EVOLUCIONAR Y LUEGO LE RESTAMOS 1 A LA SAWD DEL Oponente QUE FUE ATACADO X EL POKEMON ANTES DE EVOLUCIONAR.

$$\begin{aligned} b) \quad O_1 &= \lambda(x) \text{ASH.CANTIDAD DE POKEMON} := \text{ASH.CANTIDAD DE POKEMON} . \text{SUCC} \\ O_2 &= \text{ASH.CANTIDAD DE POKEMON} := \text{ASH.CANTIDAD DE POKEMON} . \text{SUCC} \\ O_3 &= [\text{CANTIDAD DE POKEMON} := \text{ASH.CANTIDAD DE POKEMON} . \text{SUCC}, \\ &\quad \text{ADOPTAR} = C(z) \lambda(x) z. \text{CANTIDAD DE POKEMON} := z. \text{CANTIDAD DE POKEMON} . \text{SUCC} \end{aligned}$$



NOTA